Annales Univ. Sci. Budapest., Sect. Comp. 12 (1991) 49-57

EVOLUTIONARY STRUCTURES AND MULTICRITERIA LEARNING LANGUAGES – THE RPROLOG CONCEPT

Béla Csukás and **Rozália Lakner**
University of Veszprém,
Department of Chemical Engineering Cybernetics
Veszprém, P.O. Box 158, H-8201 Hungary

Abstract: Possibilistic uncertainty associated with the multiple solutions of the Prolog programs is emphasized and a four-valued logic as well as its continuous extension for the valuation of the facts playing role in the actual proofs are discussed. The concept of the Revaluable Prolog (RPROLOG) is introduced and an illustrative example for the use of this modified language is represented. The RPROLOG is proposed for the formalization of the evolutionary learning in the solution of synthesis problems.

Keywords: Prolog, information feedback, possibilistic uncertainty, evolution, learning, synthesis problems.

1. INTRODUCTION

There are two approaches in the literature for *synthesizing the tools of fuzzy sets and fifth generation languages.*

In the *Fuzzy Relational Inference Language* (FRIL) elaborated by Baldwin and his coworkers [1], possibility and necessity values are coordinated to the relations in the sense of the possibility theory. The primary facts have a priori possibility and necessity values, while these characteristics are algorithmically considered and transformed.

In Mukaidono's approach [2] a *fuzzy resolution principle* is applied and the proposed language is based on this modified method of the Prolog's theorem provement.

In the present paper a third framework is suggested, based on the formalization of the uncertain knowledge transferred from the valuation of the alternative solutions to the generating facts building them.

The uncertain learning accompanying the solution of the synthesis problems can be described by weights in the neural nets, by fuzzy production rules, or by uncertain values coordinated to the building elements of the variants [3]. In the present work the third method is applied and the *evolutionary learning of the Prolog's solutions is based on the evaluation* of them.

2. POSSIBLE SOLUTIONS AND THEIR EVALUATION IN PROLOG

The essential feature of the Prolog-like descriptive logical languages is that the truth of the actual goal predicate is automatically deducted from the truth of the previously defined clauses. The structure of the theorem proving can be summarized by the

$$\hat{P} \Leftarrow (P) \Leftarrow \check{P} \tag{1}$$

scheme, where

$\hat{P}$ is the set of the possible solutions for a given goal predicate,

P is the set of the "intermediate" clauses, utilised for the provement of the investigated goal, and

$\check{P}$ is the set of the primary facts, which truth of the whole proof is deducted from.

Actually a single solution can be characterised by the

$$\hat{p}_i \longleftarrow (P_i) \longleftarrow \check{p}_{i,1}, \check{p}_{i,2}, \ldots, \check{p}_{i,n} \tag{2}$$

deduction scheme, where

$\hat{p}_i \in \hat{P}$ is one of the solutions,

$P_i \subset P$, and

$\check{p}_{i,j} \in \check{P}_i \subset \check{P}$ are the primary fact determining the given solution $\hat{p}_i$.

In the case of multiple solutions the Prolog's theorem prover backtrackts to *all of the possible variants systematically*. In the majority of engineering problems the alternative solutions are not equivalent, but can be *ranked or valuated* by different points of view. This ranking or valuation is often represented by the Prolog definition itself.

As an illustrative example consider the following simple Prolog definition:

```
example: -
    combination (A,B,C,D), evaluation (A,B,C,D,V1,V2).
    combination (A,B,C,D): -
      ab(A,B),bc(B,C),cd(C,D),da(D,A).
      ab(a1,b3). ab(a1,b5). ab(a1,b6)....da(d11,a9).

evaluation (A,B,C,D,V1,V2): -
```

value1(A,B,C,D,V1),value2(A,B,C,D,V2).

value1(A,B,C,D,V1): - ...
value2(A,B,C,D,V2): - ...

In this example the (a_i, b_j, c_k, d_l) combinations are derived from the ab, bc, cd and da relations, next the synthesized variants are evaluated by the value1 and value2 clauses. Suppose the combinations having better values according to these points of view are preferred.

3. A FOUR-VALUED LOGIC INDUCED BY THE TWO-VALUED EVALUATION

Let us consider temporarily only one of the above valuating points of view and assume a simple two-valued logical decision about the combinations by "true" and "false". Let us associate the value of the solutions with the true facts playing role in the proving of them as follows:

(i) At the beginning all of the possible facts are declared to be *"unknown"*.

(ii) In the knowledge of the combination's value the facts are *revaluated* by the

"unknown" $\vee$ "true" = "true"
"unknown" $\vee$ "false" = "false"
"true" $\vee$ "false" = "uncertain"
"uncertain" $\vee$ "true" = "uncertain"
"uncertain" $\vee$ "false" = "uncertain"

expressions, while the

"unknown" $\vee$ "uncertain" = "uncertain"

statement has no significance practically.

(iii) From the value coordinated to the primary facts the *possible value of the solution* can be derived by the following rules:

"unknown" $\wedge$ "uncertain" = "unknown"
"unknown" $\wedge$ "false" = "unknown"
"unknown" $\wedge$ "true" = "unknown"
"true" $\wedge$ "false" = "unknown"
"uncertain" $\wedge$ "true" = "true"
"uncertain" $\wedge$ "false" = "false"

As it can be seen from the enumerated rules, the traditional logical valuation of the solutions induces a special *four-valued logic* above the respective facts. In this four-valued logic the possibilistic uncertainty of multiple solutions is represented and even *uncertain proposals* can be made for the feasible solutions.

4. A CONTINUOUS INTERVAL LOGIC INDUCED BY THE CONTINUOUS EVALUATION

Similarly to the extension of the two-valued logic to the fuzzy logic, the four-valued logic can be extended to a special *continuous interval logic*. In this case the proved theorems are valuated by one (or more) normalised functions, while the true facts utilised for the proof can be evaluated by continuous subintervals of the [0,1] interval. In this subintervals the *pessimistic* and *optimistic* values of the respective facts are contained. In addition to these limit values the various characteristics of the *value distribution function* as well as various probabilistic measures can also be applied in the more sophisticated investigations.

The *evolution of the uncertain knowledge* about a given fact is illustrated in Fig. 1. The fact ab(a1,b6) appears firstly in the 3rd solution and later in the 6th, 17th, ...etc. variants. There is a characteristic increase of the uncertainty expressed by the length of the interval.

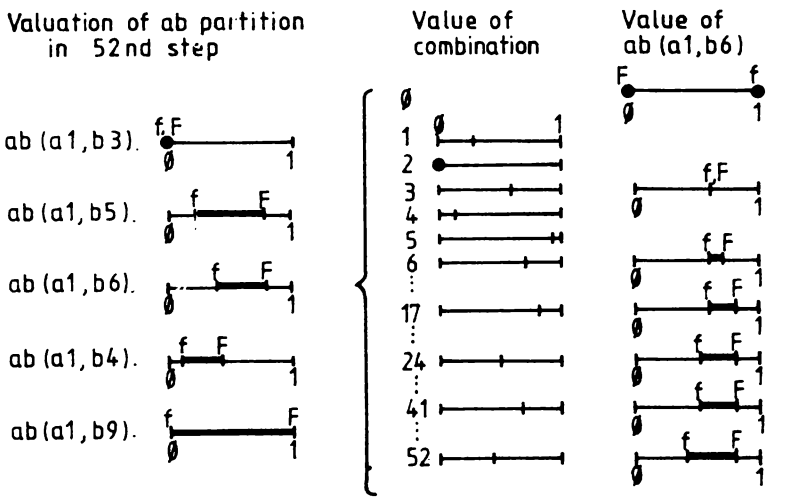

Fig. 1. Evolution of the uncertain values of a given fact

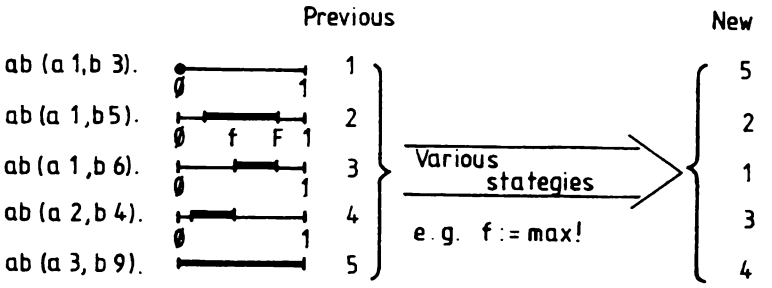

Fig. 2. Ranking of "ab" partition

Based on these uncertain values useful propositions can be made for the *feasible ranking* of the generating facts. For example in Fig. 2 the pessimistic strategy is illustrated, where the selection in governed by the maximisation of the pessimistic value.

Associating more than one uncertain valuations to the generating fact the *multicriteria situations* can also be easily formalised without any previous aggregation of the individual goal functions or constraints. The multicriteria good solutions either can be selected *a posteriori* from the intersection of the sets containing the partially good variants, or *fictitious valuations expressing the consensus* or conflict of the points of view can be derived (Fig. 3.)

5. CONCEPT OF REVALUABLE PROLOG (RPROLOG)

The *Revaluable Prolog* (RPROLOG) is a new approach for the formalisation of the possibilistic uncertainty in the framework of the logical programming. In RPROLOG the Prolog's ability for the determination of all possible variants remains in principle, however, instead of the systematic search, *the successive solutions tend to satisfy the previously declared valuating points of views.*

In RPROLOG there are various *kinds of predicates,* namely:

ordinary predicates,

valuated predicates,

evaluating predicates and

revaluable predicates.

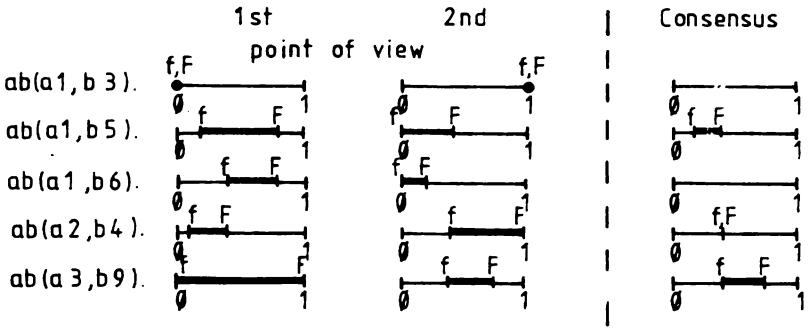

Fig. 3. Example for the consensus of two valuation

The *valuated predicates* may have multiple solutions which are evaluated by one or more points of view, described by the related *evaluating predicates*. The *revaluable predicates* represent a significant part of the predicates playing role in the proof of the connected valuated predicate. The revaluable predicates are supplemented by an uncertain valuation e.g. in the sense of the previously described continuous interval logic. The connected ensemble of a valuated predicate with the related evaluating and revaluable predicates is called *cybernetical module*. The same predicate can be valuated predicate in one module and revaluable or evaluating predicate in the other.

In the graphical illustration of the cybernetical modules (see Fig. 4) the rectangles correspond to the valuated and revaluable predicates, while the circles refer to the evaluating predicates. The triangles symbolize the operation of the *metainterpreter,* as well as the effect of the uncertain valuations on the execution of the logical program.

In RPROLOG the cybernetical modules can be *directly declared* by a special dynamic database.

The *run of the RPROLOG programs* can be characterized by the following features:

Having selected a solution for the valuated clause of the module, the program *looks for the evaluating clauses* belonging to it and evaluates the solution according to each point of view.

- Next the metainterpreter searches for the revaluable clauses of the given module and *revaluates the uncertain values of the utilized true clauses.*

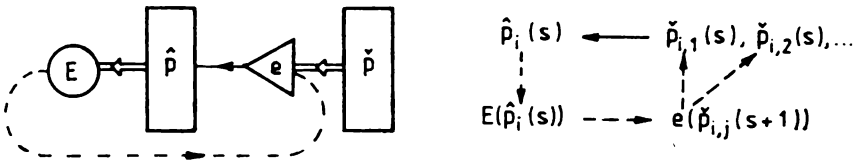

Fig. 4. Scheme of a cybernetical module

-Finally the classes or *partitions of the revaluable clauses are modified* (e.g. rearranged), and the next solution will be deduced on the basis of the modified clauses.

For example the cybernetical module of our example can be described by the

```
cyb_module("example", ["ab","bc","cd","da"],
                ["value1","value2"]).
```

fact, while the respective program using the RPROLOG's standard predicates can be written as follows:

```
example: -
  open_cyb_modul("example")
  combination(A,B,C,D),
  evaluation(A,B,C,D,V1,V2),
  close_cyb_module("example").

  combination(A,B,C,D): -
    search("example","ab",ab(A,B)),
    search("example","bc",bc(B,C)),
    search("example","cd",cd(C,D)),
    search("example","da",da(D,A)),

evaluation(A,B,C,D,V1,V2): -
  valuate("example","value1",value1(A,B,C,D,V1)),
  valuate("example","value2",value2(A,B,C,D,V2)).
```

6. TOWARDS THE FORMALIZATION OF EVOLUTIONARY STRUCTURES AND EVOLUTIONARY LEARNING

The RPROLOG seems to be an adequate tool for the investigation of the various cybernetical structures.

Cybernetical structures are characterised by the *information feedback*. In the *evolutionary structures* the information feedback means *learning from the value of the tried-solutions*. The evolutionary structures are built from so-called cybernetical modules.

A few examples for various evolutionary structures are illustrated in Fig. 5.

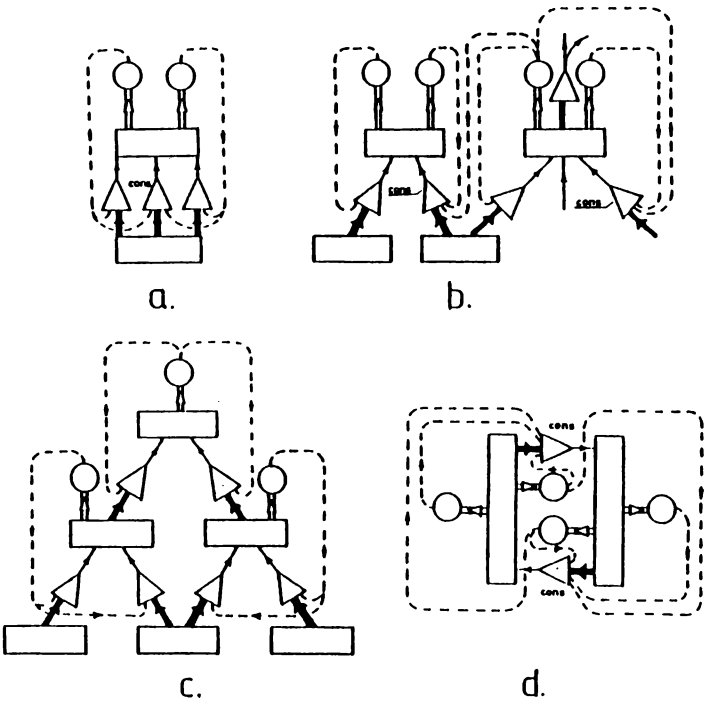

Fig. 5. Cybernetical structures

REFERENCES

[1] J.F. Baldwin, A fuzzy relational inference languages for expert systems, International Symposium on Multi-valued Logic (Japan, 1983).

[2] Y. Shen, L. Ding and M. Mukaidono, A theoratical framework of Fuzzy Prolog Machine, In: M. Gupta and T. Yamakawa Eds., *Fuzzy Computing: Theory, Hardware and Applications*, (North-Holland, Amsterdam, 1988), 89-100.

[3] P. Árva and B. Csukás, Synthesis of engineering objects by recurcive fuzzy valuation of crisp combinations, *Fuzzy Sets and Systems*,**32** (1989), 13-33.